A SAME NET SPACING VIOLATION CHECKER

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention generally relates to methods and systems for checking spacing of wiring in a semiconductor structure and more particularly to a method and system for checking the spacing of wiring within a single net.

Description of the Related Art

Conventional design rules require wires of the same net within a semiconductor structure to be spaced a specified minimum distance apart. Older designs satisfied the minimum spacing requirements fairly easily through simplified grid designs which automatically guaranteed that minimum spacing design rules were met. However, with advancing technologies wider wires are more prevalent. Such wide wires have larger spacing requirements than the older narrower wires. Therefore, a simple gridded solution is no longer effective with today's current wide wires.

Minimum spacing violations were conventionally recognized during the shapes processing performed by Design Rules Check (DRC). Design Rules Check is an expensive and time consuming process which is usually run after the

conductors, and determining that the possible non-diagonal error rectangle is not a true error when the possible non-diagonal error rectangle is completely covered by the third conductor.

The process of forming minimum spacing rectangles comprises forming the minimum spacing rectangles to have sides which are a minimum spacing design constraint distance from sides of respective ones of the conductor rectangles.

The conductors are preferably within a single net. If the circuit comprises a plurality of nets the process further includes checking for shorts between different ones of the nets.

The invention can also include dividing the possible error rectangle into at least two possible error rectangle if the possible error rectangle is partially covered by a third conductor of the conductors.

The invention is superior to conventional systems because the invention allows same net spacing errors to be recognized during physical design prior to Design Rules Check. The software supporting the invention performs orders of magnitude faster than the Design Rules Check solution. As such, the invention dramatically decreases the turn-around time of physical design, providing a fast solution which is available prior to final layout release.

additional metal shapes which removes the spacing violation;

Figure 10 is a revised schematic illustration of Figure 8 illustrating additional metal shapes which do not remove the spacing violation;

Figure 11 is a schematic diagram of a computer system and software program for performing the invention; and

Figure 12 is a flowchart illustrating an embodiment of the invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS OF THE INVENTION

Referring now to the drawings, and more particularly to Figure 1, a rectangle representing a conductor shape M1, such as a metal wiring, and a spacing rectangle S1, which defines a minimum space "d" according to design rules around the metal shape M1 are illustrated. The metal shape is a part of a larger overall conductive net within a structure, such as a semiconductor. The metal shape comprises circuit components which operate by transmission of signals through the net.

A first stage of the invention involves a plane sweep of all the net's components (e.g., net parts) such as vias, wires segments, pins or powers. The first rectangle M1 represents the shape of the net part and the second rectangle S1 represents the minimum spacing requirement surrounding the net part M1. The size of the second minimum spacing rectangle S1 is determined by the design

removes the possible spacing error rectangle from the possible error list.

For example, as illustrated in Figure 5, metal shape M3 completely covers possible error P1. Therefore, since no actual space exists between metal shapes M1 and M2 , the possible error rectangle P1 illustrated in Figure 5 is not a true spacing error and is properly removed from the possible error list.

Figure 6 illustrates a metal shape M3 which only partially covers the possible spacing error in P1. In such a situation, the invention creates multiple possible spacing error rectangles (e.g. P2, P3) from the original possible spacing error P1, as illustrated in Figure 7. The possible spacing error P1 is removed from the possible error list and the newly created possible spacing errors (e.g. P2, P3) are added to the possible error list. Each of the newly created possible spacing errors (e.g. P2, P3) will subsequently be evaluated to determine if the new possible spacing error rectangle is totally covered by other metal in the same fashion.

The process for determining whether a non-diagonal possible spacing errors is an actual error is discussed above. For diagonal possible spacing errors different techniques are utilize and are discussed below with respect to Figures 8-10.

More specifically, Figure 8 illustrates a diagonal possible spacing error P1 between metal shapes M1 and M2. The diagonal measure of rectangle P1 is less than the minimum spacing constraint of the design rules and no other metal shapes in the design intersect the P1 rectangle. Therefore, the possible error

invention works much faster because the complexity and number of the rectangles analyzed is reduced.

While the overall methodology of the invention is described above, the invention can be embodied in any number of different types of systems and executed in any number of different ways, as would be known by one ordinarily skilled in the art. For example, as illustrated in Figure 11, a typical hardware configuration of an information handling/computer system in accordance with the invention preferably has at least one processor or central processing unit (CPU) 11. The CPUs 11 are interconnected via a system bus 12 to a random access memory (RAM) 14, read-only memory (ROM) 16, input/output (I/O) adapter 18 (for connecting peripheral devices such as disk units 21 and tape drives 40 to the bus 12), user interface adapter 22 (for connecting a keyboard 24, mouse 26, speaker 28, microphone 32, and/or other user interface device to the bus 12), communication adapter 34 (for connecting an information handling system to a data processing network), and display adapter 36 (for connecting the bus 12 to a display device 38).

A flowchart of the foregoing embodiment of the invention is shown in Figure 12. More specifically, in block 10 the metal and space rectangles are formed for all metal shapes within the net, as discussed above. In block 20 the rectangles are compared using a plane sweep algorithm to determine which metal and space rectangles are intersecting. From this information, block 30 creates the initial list of possible spacing errors.

computer program, could determine whether a potential spacing error exists and whether the possible spacing error is diagonal or non-diagonal. An example of the pseudo-code for such a software program follows.

```
activeList = empty
netPartList = empty
possibleErrorList = empty
diagonalPossibleErrorList = empty

For each net part
     add space rectangle keyed by it's low x
          coordinate to netPartList
     add space rectangle keyed by it's high x
          coordinate to netPartList
endFor

Sort netPartList

// Iterate through netPartList
while (sortedNetPart = getNext(netPartList))
     if (sortedNetPart key is low x)
          For every activeListNetPart in activeList
*See NOTE
               Compare(sortedNetPart,activeListNetPart)
          endFor
          add sortedNetPart to activeList
     else
          delete sortedNetPart from activeList
endWhile


Compare(netPart1, netPart2)
     if netPart1 and netPart2 metal rectangles do not
          intersect
          if netPart1 space rectangle intersects
               netPart2 metal rectangle
                         OR
          netPart2 space rectangle intersects netPart1
               metal rectangle
               Compute distance between metal shapes
               if distance < spacing requirement
                    Add intersection of the two space
                         rectangles to either
                         possibleErrorList or
                         diagonalPossibleErrorList
```

```
            endIf
        endIf
    endIf
endCompare
```

```
Note: Rather than a simple iteration of the activeList
in the loop above, many other possible data structure
implementations could be applied to the activeList and
the compare stage of the activeList processing.  A
radix search tree or priority search tree could improve
performance if the number of net parts was very large.
```

Additionally, a computer program, or a portion of a computer program, could evaluate a possible non-diagonal spacing error to determine whether a possible non-diagonal error is actually a true error to be reported to the user.

More specifically, such a computer program or portion of a computer program could determine if a possible spacing error rectangle is covered by other metal in the design. The invention could remove a spacing error from the list of possible errors upon determining the possible spacing error rectangle is entirely covered by metal. Upon determining the possible spacing error rectangle is partially covered by metal, the invention could calculate new possible spacing error rectangles by subtracting the covered area from the initial possible spacing error rectangle and could replace the original possible spacing error with the newly created possible spacing errors. An example of the pseudo-code for such a software program follows.

```
// Iterate through net parts, stopping if
// possibleErrorList is empty
For each netPart and possibleErrorList is not empty
```

```
        // Iterate through possibleErrorList
        while (possibleError = getNext(possibleErrorList))
             If possibleError was not created from this
                  netPart
                  Intersect the netPart metal rectangle
                       with the possibleError rectangle
                  If the intersection rectangle is more
                       than a line
                       Subtract the intersection rectangle
                            area from the possibleError
                       rectangle, possibly
                            creating/deleting
                            possibleErrorList elements
                  endIf
             endIf
        endWhile
   endFor

   // Any remaining possibleErrorList elements are errors
   while (possibleError = getNext (possibleErrorList))
        Report possibleError rectangle as an error
   endWhile
```

Additionally, a computer program, or a portion of a computer program, could evaluate a possible diagonal error to determine whether a possible diagonal error is actually a true error to be reported to the user.

More specifically, such a computer program or a portion of a computer program could evaluate whether additional metals exists which collectively intersect the edges of the diagonal possible spacing error rectangle and whether those edge intersections indicate a metal connection between the original metal rectangles comprising the possible spacing error.

The invention could compare the diagonal possible spacing error rectangle edges to the metal rectangles of all other net parts. As metal rectangles are found which intersect the edges, the invention could split, shorten or remove the edges of the possible error rectangle at the area of the intersection of the possible error rectangle and the metal rectangle. The invention could then evaluate the remaining edges to determine if two adjacent edges connecting the original metal shapes comprising the diagonal possible spacing error have been removed by this process and, if so, could remove the possible spacing error from the list of possible spacing errors. This situation is similar to that shown in Figure 9.

However, if the edges of the possible diagonal error rectangle which have been removed do not connect the original metal shapes comprising the error, the possible spacing error would be considered a true spacing violation. This is similar to the situation shown in Figure 10. An example of the pseudo-code for such a software program follows.

```
// Iterate through diagPossibleErrorList
while (possibleError = getNext(diagPossibleErrorList)

     if possibleError rectangle is a line

          create one edge

          // Iterate though net parts, stopping if
          // edge is null
          For each netPart and edge != NULL
               process_this_edge(netPart, edge)
          endfor

          if (edge == NULL)  // Edge totally removed
               remove possibleError from
               diagPossibleErrorList
          endif
```

```
        else  // possibleError is a rectangle

              create four edges (east, west, north, south)
              still_error = true

              // Iterate through net parts stopping if
              // no longer an error
              For each netPart and still_error
                   process_this_edge(netPart, edge_east)
                   process_this_edge(netPart, edge_west)
                   process_this_edge(netPart, edge_north)
                   process_this_edge(netPart, edge_south)
              endfor

              // Have the appropriate edges been removed?
              if ((metal shapes are at the NE and SW
                   corners of possibleError rectangle) AND
                  ((edge_east  == NULL) &&
                   (edge_south == NULL)) OR
                  ((edge_west  == NULL) &&
                   (edge_north == NULL)))
                   still_error = false
              endif

              if ((metal shapes are at the NW and SE
                   corners of possibleError rectangle) AND
                  ((edge_west  == NULL) &&
                   (edge_south == NULL)) OR
                  ((edge_east  == NULL) &&
                   (edge_north == NULL)))
                   still_error = false
              endif

              if ( ! still_error )
                   remove possibleError from
                   diagPossibleErrorList
              endif

        endif
endwhile


// Any remaining diagPossibleErrorList elements are
// errors
while (possibleError = getNext(diagPossibleErrorList))
     Report possible Error rectangle as an error
endWhile
```

```
process_this_edge(netPart, edge_list)

      // Iterate through partial edges associated with
      // this edge
      while (partial_edge = getNext(edge_list))
            if netPart metal rectangle intersects
            partial_edge
                  if intersection is entire partial edge
                        remove partial_edge from edge_list
                  else
                        modify partial edge to reflect
                        remaining edge after intersection
                        possibly splitting into two
                        partial edges
                  endIf
            endIf
      endWhile

endProcess_this_edge
```

Therefore, the invention is superior to conventional systems because the invention allows same net spacing errors to be recognized during physical design prior to Design Rules Check. The software supporting the invention performs orders of magnitude faster than the Design Rules Check solution. As such, the invention dramatically decreases the turn-around time of physical design, providing a fast solution which is available prior to final layout release.

The invention is general in nature and can be applied to any application which can represent the application data as a set of connected rectangles and a set of spacing constraints. For example, any application which architects nets, such as mazes for mice or other objects to pass through, might wish to ensure the spacing in the structure would meet a minimum requirement so that the mice or other objects do not get stuck.

While the invention has been described in terms of preferred embodiments, those skilled in the art will recognize that the invention can be practiced with modification within the spirit and scope of the appended claims.